give me python code for an online voting system that is secured with blockchain technology and you can only vote once with a special voter id

ChatGPT said:

ChatGPT

Below is a Python implementation of a simple online voting system using blockchain technology. It ensures that each voter can vote only once using a unique voter ID. For the blockchain, we'll use a basic implementation to demonstrate the concept.

```
import hashlib

import json

from time import time

from flask import Flask, jsonify, request


class Blockchain:

  def __init__(self):

    self.chain = []

    self.current_votes = []

    self.voter_ids = set()  # To ensure unique voter IDs

    self.create_block(previous_hash='1', proof=100)  # Genesis block


  def create_block(self, proof, previous_hash):

    """Creates a new block and adds it to the chain."""

    block = {

      'index': len(self.chain) + 1,

      'timestamp': time(),

      'votes': self.current_votes,

      'proof': proof,

      'previous_hash': previous_hash
```

```
        }
        self.current_votes = []  # Reset the current votes
        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def proof_of_work(self, previous_proof):
        """Implements a simple proof-of-work algorithm."""
        new_proof = 1
        check_proof = False
        while not check_proof:
            hash_operation = hashlib.sha256(str(new_proof**2 -
previous_proof**2).encode()).hexdigest()
            if hash_operation[:4] == '0000':
                check_proof = True
            else:
                new_proof += 1
        return new_proof

    def hash(self, block):
        """Creates a SHA-256 hash of a block."""
        encoded_block = json.dumps(block, sort_keys=True).encode()
        return hashlib.sha256(encoded_block).hexdigest()
```

```
    def is_chain_valid(self, chain):
        """Checks if the blockchain is valid."""
        previous_block = chain[0]
        block_index = 1
        while block_index < len(chain):
            block = chain[block_index]
            if block['previous_hash'] != self.hash(previous_block):
                return False
            previous_proof = previous_block['proof']
            proof = block['proof']
            hash_operation = hashlib.sha256(str(proof**2 -
previous_proof**2).encode()).hexdigest()
            if hash_operation[:4] != '0000':
                return False
            previous_block = block
            block_index += 1
        return True


    def add_vote(self, voter_id, candidate):
        """Adds a vote if the voter ID is unique."""
        if voter_id in self.voter_ids:
            return False  # Voter ID already used
        self.voter_ids.add(voter_id)
        self.current_votes.append({'voter_id': voter_id, 'candidate': candidate})
        return True
```

```
# Flask web application
app = Flask(__name__)
blockchain = Blockchain()

@app.route('/vote', methods=['POST'])
def vote():
    json_data = request.get_json()
    required_fields = ['voter_id', 'candidate']
    if not all(field in json_data for field in required_fields):
        return 'Invalid data', 400

    voter_id = json_data['voter_id']
    candidate = json_data['candidate']

    if blockchain.add_vote(voter_id, candidate):
        response = {'message': 'Vote added successfully!'}
    else:
        response = {'message': 'Voter ID has already been used.'}

    return jsonify(response), 201

@app.route('/mine', methods=['GET'])
def mine():
    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
```

```
    previous_hash = blockchain.hash(previous_block)
    block = blockchain.create_block(proof, previous_hash)
    response = {
        'message': 'New block mined!',
        'index': block['index'],
        'timestamp': block['timestamp'],
        'votes': block['votes'],
        'proof': block['proof'],
        'previous_hash': block['previous_hash']
    }
    return jsonify(response), 200

@app.route('/chain', methods=['GET'])
def get_chain():
    response = {
        'chain': blockchain.chain,
        'length': len(blockchain.chain)
    }
    return jsonify(response), 200

@app.route('/validate', methods=['GET'])
def validate():
    is_valid = blockchain.is_chain_valid(blockchain.chain)
    response = {'message': 'Blockchain is valid!' if is_valid else 'Blockchain is not valid!'}
    return jsonify(response), 200
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Features of the Code:

1. **Blockchain Implementation**:
    - Each block stores votes, a proof of work, and a hash of the previous block.
    - Uses a simple proof-of-work algorithm to secure the blockchain.
2. **Unique Voter IDs**:
    - A set (voter_ids) ensures each voter can only vote once by checking their voter ID.
3. **REST API**:
    - /vote: Endpoint to cast a vote using a voter_id and candidate.
    - /mine: Mines a new block to record the votes securely.
    - /chain: Retrieves the entire blockchain.
    - /validate: Validates the blockchain integrity.
4. **Flask Framework**:
    - Provides a web server to interact with the voting system.

Usage Instructions:

1. Install dependencies:

bash

Copy code

pip install flask

2. Run the server:

python blockchain_voting_system.py

1. Use a tool like Postman or curl to interact with the API.

Example Voting Request:

POST /vote

```
{
  "voter_id": "unique_voter_id_123",
  "candidate": "Alice"
}
```

This system demonstrates the fundamentals of combining blockchain with an online voting mechanism. It can be expanded for production with additional features like cryptographic signatures for voter IDs, enhanced security, and scalability.